


9. Digitale Verarbeitung 3-dimensionaler Darstellungen

- 9.1 Grundlagen der 3D-Computergrafik 
- 9.2 3D-Modellierung am Beispiel VRML
- 9.3 Interaktion in 3-dimensionalen Darstellungen
- 9.4 3D-Grafik-Programmierung
(Beispiel Java 3D)

Literatur:

Henning, Taschenbuch Multimedia, Kap. 13

Rolf Däßler: VRML - 3D-Welten im Internet, bhv Verlag 2002

<http://www.web3d.org>

Alan Watt: 3D Computergrafik, 3. Auflage, Pearson Studium 2002

Dreidimensionale Darstellung

- Dimensionenkonflikt:
 - Die reale Welt ist dreidimensional
 - Bilddarstellungen (wie bisher betrachtet) sind zweidimensional
 - » Verdeckte Ansichten und Details
- Dreidimensionale Darstellung:
 - Setzt Modell mit den Informationen in allen drei Dimensionen voraus
 - » Alle möglichen Ansichten ohne Informationsverlust
- Anwendungsbereiche für dreidimensionale Darstellung:
 - Virtuelle Welten, „Cyberspace“
 - Ingenieur Anwendungen:
 - » CAD (z.B. Maschinenbau)
 - » Designmodelle von Produkten
 - » (Interaktive) Architekturmodelle
 - Produktpräsentation
 - Geovisualisierung
 - Animation im Film (Trickfiguren in klassischem Film, Vollanimation)

3D-Eingabegeräte

- Zeigergeräte: Siehe früheres Kapitel der Vorlesung
- 3D-Scanner:
 - Abtastung realer Objekte z.B. mit Ultraschall, Laser
- Regelmässige Abtastung von 2D-Daten:
 - Z.B. Bildgebende Verfahren der Medizin
 - » Computertomographie (Röntgenbilder)
 - » Magnetresonanztomographie (Kernspin-Prinzip)
 - Errechnung eines „Volumendatensatzes“
 - » Nicht nur Aussenansicht, sondern Information über alle möglichen Schnitte
 - » Filterung anhand verschiedener Kriterien, z.B. Farb-Schwellwerte



Indeed Amira

3D-Ausgabegeräte

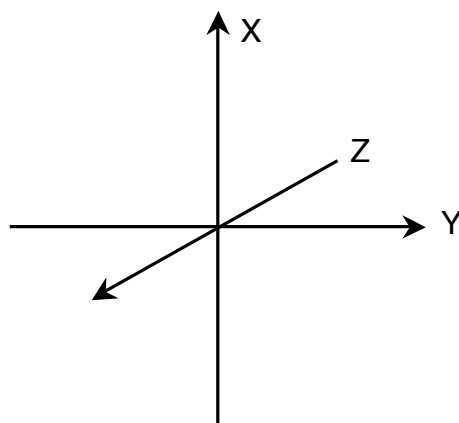
- Echte 3D-Darstellung:
 - 3D-Drucker (erzeugt Gegenstände)
- Stereo-Vision:
 - Separate Bildinformation für die beiden Augen
 - Polarisations- und Farbbrillen
 - Shutterbrillen (abwechselnd ein Auge abgedunkelt)
- Immersive Verfahren:
 - *Head Mounted Display (HMD)*
 - Spezialräume mit Rundum-Projektion (*Cave*)
 - Anpassung der Darstellung an Position und Bewegung des Betrachters durch *tracking*
- Einfachste Darstellungsform:
 - 2D-Rendering in „Fenster zur Welt“
 - 2D-Darstellung wird flexibel aus 3D-Daten berechnet



Virtual Reality (VR)

- Oberbegriff für alle Techniken, die die natürliche Wahrnehmung der Umgebung über die Sinnesorgane simulieren
 - Idealvision: „Holodeck“ der USS Enterprise
 - Räumliche, realistische Wahrnehmung von Gegenständen
 - Räumliche, mit der Bildwahrnehmung integrierte Tonwahrnehmung
 - Natürlich Interaktion mit der Kunstwelt, z.B. durch Gehen
- Begriff VR für einfachere Technologien verwendet:
 - Z.B. Modellierung von 3D-Welten mit 2D-Rendering („VRML“)
- Augmented Reality (AR):
 - Ergänzung der realen Welt um virtuelle Elemente
 - Z.B. Einblendung künstlich erzeugter Elemente in reale Gegenstände
 - Z.B. Informationsaustausch über reale Gegenstände

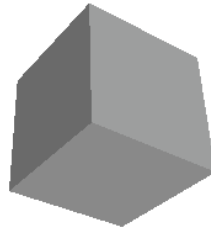
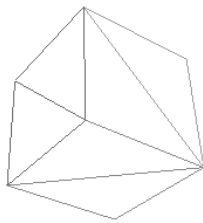
3D-Koordinatensystem



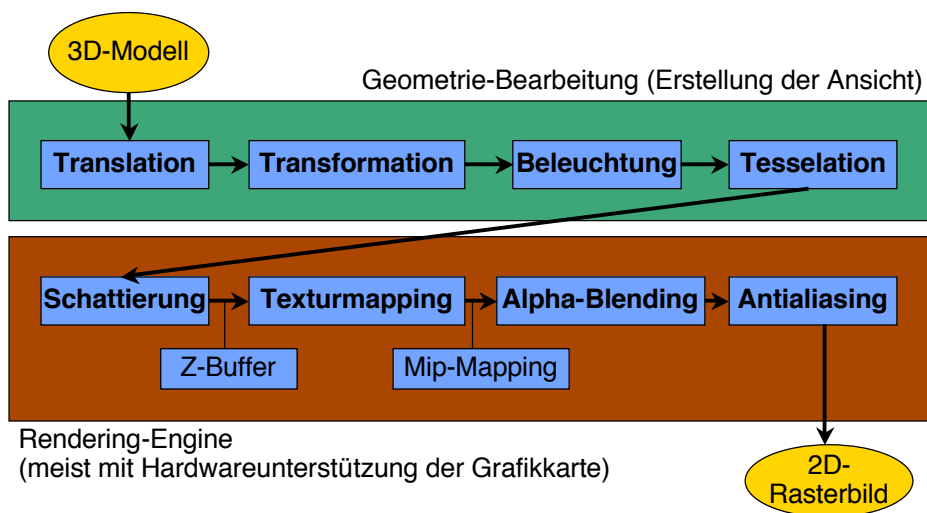
Kartesisches Koordinatensystem
Merkhilfe: „Rechte-Hand-Regel“

Grundidee der 3D-Modellierung

- Gegenstände:
 - Punktwolken im 3-dimensionalen Raum
 - Zusatzinformationen z.B. zur Oberflächenstruktur
- Verbindung der Punkte in definierter Weise:
 - » Rendering als *Drahtmodell*
- Anpassung des Rendering an visuelle Wahrnehmung:
 - Perspektive, Verdeckung



3D-Rendering-Pipeline



Translation

- Übersetzung der Modellkoordinaten in den zum Rendering benutzten Koordinatenraum
 - Integration von Modellen aus verschiedenen Quellen, z.B. verschiedenen Entwicklungssystemen
 - Häufig: *Weltkoordinatensystem*
- *Clipping*
 - Abschneiden von Objektteilen ausserhalb des Blickwinkels des Beobachters

Transformation

- Änderung der Objektposition
 - Verschieben (oft auch *translation* genannt)
 - In 3 Freiheitsgraden
- Änderung der Objektausrichtung
 - Rotation
 - In 3 Freiheitsgraden
- Änderung der Objektgrösse
 - Skalierung
 - 1 Freiheitsgrad
- Bewegung der Betrachterposition in einer virtuellen Welt:
 - Obige Operationen treten (kombiniert) extrem häufig auf
 - Schnelle Implementierung wichtig

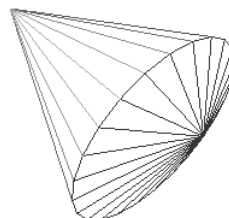
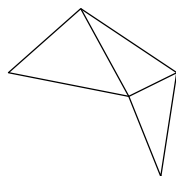
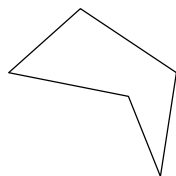
Beleuchtung

- Einfluss von Lichtquellen auf das Erscheinungsbild einer 3D-Szene
 - Ganz ohne Lichtquellen: Schwarz!
- Abhängig von:
 - Standort und Art von Lichtquellen
 - Spezialfall einer Standard-Lichtquelle:
 - » „Headlight“ aus der Richtung des Betrachter
 - Standpunkt und Blickrichtung des Betrachters



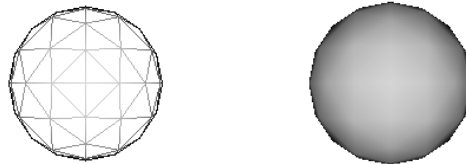
Tessellation

- Durch 3D-Rendering nur Polygone darstellbar!
 - Komplexe Szenen aus extrem vielen (Millionen) von Polygonen zusammengesetzt
- Darzustellendes Objekt wird in einfache Polygone (meist Dreiecke) zerlegt
- Tessellation = Zerlegung komplexer Polygone in Dreiecke



Schattierung (*shading*)

- *Flat-Shading*:
 - Berechnet für jedes Flächenelement (Polygon) der 3D-Szene einen Helligkeitswert
 - Bestimmt sich aus der Winkeldifferenz zwischen einfallendem Licht und dem Normalenvektor des Polygons
 - Einfach zu berechnen
 - Nachteil: Homogener Farbwert je Polygon
- Verfeinerte Schattierungsverfahren:
 - Z.B. Gouraud-Shading
 - Interpolation der Farbwerte mit benachbarten Polygonen
 - Erreicht relativ gute optische „Glättung“



Z-Buffer

- Z-Buffer speichert für jeden Bildpunkt des 2D-Bildes die niedrigste Entfernung zu einem Objekt
- Beschleunigung des Rendering:
 - Offensichtlich verdeckte Objekte bzw. Objektteile müssen nicht betrachtet werden
- Größere Wahlfreiheit bei der Abarbeitung des Rendering
 - Hintergrundbild muss nicht unbedingt zeitlich vor den Vordergrundobjekten gerendert werden
- Hardwareunterstützung in Grafikkarten

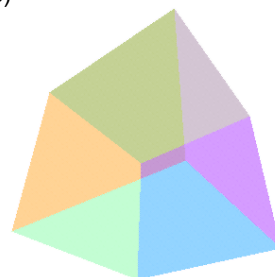
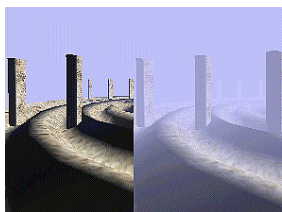
Textur-Mapping

- Textur:
 - Muster oder Bild, das auf Oberflächen von 3D-Objekten gelegt wird
 - Kann oft Anzahl der benötigten Polygone drastisch reduzieren
- Textur-Mapping:
 - Darstellung der Flächen eines Objekts mit Textur
 - Erweiterung: Perspektivische Korrektur der Textur
- Mip-Mapping:
 - Textur in mehreren Auflösungen verfügbar (und automatisch passende Fassung ausgewählt)




Alpha-Blending

- Kontrolle der Transparenz eines Objekts
 - Analog zu 2D-Rendering
- Bei 3D-Grafik:
 - Tiefeneindruck durch „Verwischen“ von Details bei grösserer Entfernung
 - Nebeneffekt (*fogging*)
 - „Depth cueing“ (Überblendung ins Schwarze)



9. Digitale Verarbeitung 3-dimensionaler Darstellungen

- 9.1 Grundlagen der 3D-Computergrafik
- 9.2 3D-Modellierung am Beispiel VRML 
- 9.3 Interaktion in 3-dimensionalen Darstellungen
- 9.4 3D-Grafik-Programmierung (Beispiel Java 3D)

Literatur:

Henning, Taschenbuch Multimedia, Kap. 13

Rolf Däßler: VRML - 3D-Welten im Internet, bhv Verlag 2002
http://www.web3d.org/resources/vrml_ref_manual/Book.html

Virtual Reality Modeling Language VRML

- Skriptsprache zur Beschreibung von 3D-Welten
 - Auf den Einsatz im Internet ausgelegt
 - Universelles Austauschformat für 3D-Szenen
- Geschichte:
 - Basiert auf Grafikstandard „OpenInventor“ von Silicon Graphics
 - Marc Pesce, Toni Parisi, 1994: Erster 3D-Browser, Entwurf VRML 1.0
 - April 1995: VRML Version 1.0 verabschiedet
 - Konkurrierende Vorschläge für die Weiterentwicklung, insbesondere für Dynamik
 - 1996: Internet-Abstimmung über VRML 2.0, gewonnen von *MovingWorlds*-Standard (Silicon Graphics & Sony), VRML 2.0 verabschiedet
 - 1997: VRML wird Internationaler Standard ISO-14772
 - » Meist als „VRML 97“ bezeichnet
 - » Weitgehend identisch zu VRML 2.0
- Dateierweiterung:
 - .wrl (wie „world“)

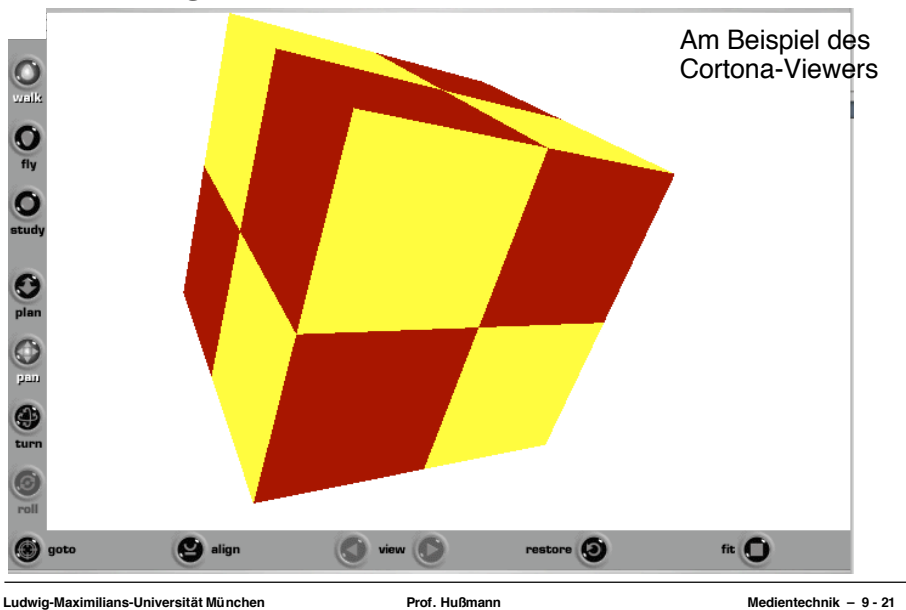
Einordnung von VRML

- VRML ist ein Grafik-Dateiformat
- VRML ist ein Vektor-Grafikformat
- VRML spielt eine ähnliche Rolle wie SVG für 2D-Grafik
- VRML hat allerdings *keine* XML-Syntax!

Softwarewerkzeuge für VRML

- Anzeigeprogramme (*viewer*)
 - Meist als „Plug-In“ für Web-Browser
 - Bekannte Produkte:
 - » *CosmoPlayer* (Silicon Graphics)
 - » *Cortona* (Parallel Graphics)
 - » *FreeWRL* (OpenSource-Aktivität)
- Autorenwerkzeuge:
 - Einfache syntaxunterstützende Editoren (z.B. VrmIPad)
 - Spezielle 3D-Editoren
 - Aufwändige 3D-Modellierungs- und Animationswerkzeuge mit VRML-Exportfunktion
 - » Z.B. 3d studio max, SoftImage, Maya, Cinema4D

Bedienungselemente eines VRML-Viewers



Navigationsmodi

- Grundmodi:
 - Walk: Bewegung des Betrachters nur in der horizontalen Ebene
 - Fly: Bewegung des Betrachters auch in der vertikalen Ebene
 - Study: Bewegung der Welt um das Zentrum des Objekts
- Optionen (in Kombination mit den Grundmodi):
 - Plan: Bewegungseingaben beziehen sich auf Verschiebung in der horizontalen Ebene
 - Pan: Bewegungseingaben beziehen sich auf Verschiebung in der vertikalen Ebene
 - Turn: Bewegungseingaben beziehen sich auf Drehung in der horizontalen Ebene
 - Roll: Bewegungseingaben beziehen sich auf Drehung in der vertikalen Ebene
- Bewegungseingaben erfolgen z.B. durch Pfeiltasten oder Mausgesten

Syntax von VRML

- Bezeichner empfindlich gegen Gross- und Kleinschreibung!
- Knoten:
 - *Knotentypbezeichner* { *Attribute* }
 - Knotentypbezeichner beginnt immer mit Grossbuchstaben
 - Z.B. `Sphere { radius 1.0 }`
- Felder:
 - Folgen von Paaren *Feldtypbezeichner* *Feldwert*
 - Feldtypbezeichner beginnt immer mit Kleinbuchstaben
 - Z.B. `radius 1.0`
- Listen von Werten:
 - Z.B. `[0, 1, 2, 3, 4]`
- Datentypen:
 - Ganze Zahlen, reelle Zahlen, Zeichenketten, Boolesche Werte u.v.a.
- Einheiten:
 - VRML-Einheiten müssen extern interpretiert werden, z.B.
Längeneinheit = Meter, Winkleinheit = rad, Zeiteinheit = Sekunde

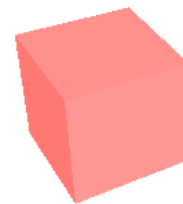
VRML-Beispiel: box0.wrl

```
#VRML V2.0 utf8
Background { skyColor 1.0 1.0 1.0 }

Shape {

  appearance Appearance {
    material Material {
      emissiveColor 1.0 0 0
    }
  }

  geometry Box {
    size 2.0 2.0 2.
  }
}
```



Shape-Knoten

- Knotentyp **Shape**
 - Benötigt Felder **appearance** und **geometry**
- Feldtyp **appearance**
 - Enthält in der Regel einen Knoten vom Typ **Appearance**
 - » Angabe diverser Materialeigenschaften (Farbe, Schattierung, ...)
- Feldtyp **geometry**
 - Enthält Geometrie-knoten
- Übersicht wichtiger Geometrie-knotentypen:
 - **Box**: Quader (**size**)
 - **Cone**: Kegel (**bottomRadius**, **height**)
 - **Cylinder**: Zylinder (**radius**, **height**)
 - **Sphere**: Kugel (**radius**)
 - **Text**: 3D-Text
 - ...

Hintergrund

- Grundkonzept:
 - Halbkugel mit unendlichem Radius als Boden (*ground*)
 - Kugel mit unendlichem Radius als Himmel (*sky*)
- **Background**-Knoten:
 - Spezifikation der Boden- und Himmelfarben
 - » **groundColor**, **skyColor**
 - Möglichkeit der Beschreibung von Abstufungen
 - » Liste von Farben und Winkel, in denen sie angewandt werden
 - Möglichkeit der Einbindung von Texturen

Benutzerdefinierte Formen

- Beliebige Formen können über Koordinatenwerte definiert werden
 - Knotentyp `Coordinate`, Feldtyp `point`, Werte 3er-Gruppen von reellen Zahlen
- Bildung von Objekten mit `IndexedLineSet` bzw. `IndexedFaceSet`:
 - `IndexedLineSet` erzeugt Gittermodell, `IndexedFaceSet` Flächenmodell
 - Feld `coord` enthält die beteiligten Punkte
 - » Implizit werden die Punkte, mit 0 beginnend, nummeriert (je drei Zahlen = 1 Punkt)
 - Feld `coordIndex` enthält die einzelnen anzuzeigenden Linien bzw. Flächen
 - » Als Indizes in der Punktliste
 - » Jedes Element (Linie bzw. Fläche) mit -1 abgeschlossen

Beispiel: Würfel selbstdefiniert (Drahtgitter)

```
Shape {
  appearance ...
  geometry IndexedLineSet {
    coord Coordinate {
      point [
        -1.0 1.0 1.0, # Punkt 0: links oben vorn
        -1.0 -1.0 1.0, # Punkt 1: links unten vorn
        1.0 -1.0 1.0, # usw.
        1.0 1.0 1.0,
        -1.0, 1.0, -1.0,
        -1.0, -1.0, -1.0,
        1.0, -1.0, -1.0,
        1.0, 1.0, -1.0
      ]
    }
    coordIndex [
      0, 1, 2, 3, 0, -1, # vorderes Quadrat
      4, 5, 6, 7, 4, -1,
      0, 4, -1,
      1, 5, -1,
      2, 6, -1,
      3, 7
    ]
  }
}
```

Beispiel: Würfel selbstdefiniert (Flächen)

```
Shape {
  appearance ..
  geometry IndexedFaceSet {
    solid FALSE
    coord Coordinate {
      point [... wie oben ...]
    }
    coordIndex [
      0, 1, 2, 3, 0, -1,
      4, 5, 6, 7, 4, -1,
      0, 3, 7, 4, 0, -1,
      1, 2, 6, 5, 1, -1,
      3, 2, 6, 7, 3, -1,
      0, 1, 5, 4, 0
    ]
  }
}
```

Appearance- und Material-Knoten

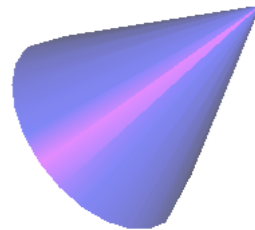
- Knotentyp **Appearance**
 - Optionale Felder **material**, **texture** und **textureTransform**
- Feldtyp **material**
 - Enthält in der Regel einen Knoten vom Typ **Material**
- Feldtyp **texture**
 - Enthält einen Texturknoten (siehe unten)
- Feldtypen im Materialknoten:
 - (Werte immer zwischen 0.0 und 1.0)
 - **ambientIntensity**: Reflexion für Umgebungslicht
 - **diffuseColor**: Reflektierende (nicht leuchtende) Farbe
 - **emissiveColor**: Selbstleuchtende Farbe
 - **shininess**: Stärke von Glanzlichtern
 - **specularColor**: Farbe von Glanzlichtern
 - **transparency**: Durchsichtigkeit
- Spezifikation von Farben:
 - als RGB-Wert (Zahlentripel)

Beispiel: Materialeigenschaften

```
#VRML V2.0 utf8
Background { skyColor 1.0 1.0 1.0 }

Shape {
  appearance Appearance {
    material Material {
      diffuseColor 0.2 0.2 1.0
      shininess 1.0
      specularColor 1.0 0 0
      transparency 0.3
    }
  }

  geometry Cone {
    bottomRadius 1.0
    height 2.0
  }
}
```



Texturen

- Knotentyp **ImageTexture**
 - Benötigt Feldtyp **url** zur Angabe einer Datei mit 2D-Grafik (JPEG, PNG, GIF)
 - Achsen des Texturbildes mit S (horizontal) und T (vertikal) bezeichnet
 - Feldtypen **repeatS**, **repeatT** (Boolean) zur Steuerung der Wiederholung
- Knotentyp **PixelTexture**
 - Direkte Angabe einer Textur als Pixelfeld in VRML
- Knotentyp **MovieTexture**
 - Analog zu **ImageTexture**, aber mit Bewegtbild (MPEG-1)
 - Zusätzliche Feldtypen:
loop, **speed**, **startTime**, **stopTime**

Beispiel: Quader mit Textur

```
#VRML V2.0 utf8

Background { skyColor 1.0 1.0 1.0 }

Shape {

  appearance Appearance {
    texture ImageTexture {
      url "textur0.gif"
    }
  }

  geometry Box { }

}
```

Szenegraphen: Group- und Transform-Knoten

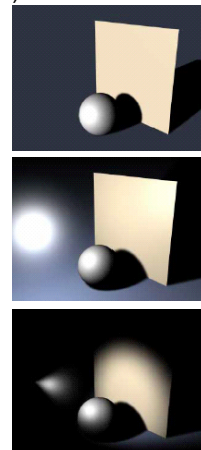
- Ein *Szenegraph* ist eine Baumstruktur, die alle in einer 3-dimensionalen virtuellen Welt enthaltenen Objekte mit ihren Eigenschaften enthält
- Wurzel des Szenegraphen: **Group**-Knoten
 - enthält Liste von Objekten im **children**-Feld
- Darstellung an anderer Stelle als im Ursprung durch **Transform**-Knoten
 - Anwendung von Transformationen *in folgender Reihenfolge*
 - **children**-Feld gibt Knoten an, die transformiert werden
 - **center**-Feld: Definition eines neuen Mittelpunkts
 - **rotation**-Feld: Drehung um Winkel
 - » Angabe in rad
 - » Tripel: x-Achse (*pitch*), y-Achse (*yaw*), z-Achse (*roll*)
 - » Positives Vorzeichen bedeutet Rechtsdrehung
 - **scale**-Feld: Maßstäblich veränderte Darstellung
 - **translation**-Feld: Verschiebung um Vektor

Beispiel: Einfacher Szenegraph

```
Group {
  children [
    Transform {
      children [
        Shape {
          appearance Appearance {
            material Material {
              diffuseColor 1.0 0 0
            }
          }
          geometry Box {
            size 2.0 2.0 2.0
          }
        }
      ]
      translation 2.0 0 0
    }
    Shape {
      appearance Appearance {
        material Material {
          diffuseColor 0 0 1.00
        }
      }
      geometry Sphere {
        radius 1.0
      }
    }
  ]
  ... (rechte Spalte)
}
...
Transform {
  children [
    Shape {
      appearance Appearance {
        material Material {
          diffuseColor 0 1.0 0
        }
      }
      geometry Box {
        size 2.0 2.0 2.0
      }
    }
  ]
  translation -2.0 0 0
}
NavigationInfo {
  type "EXAMINE"
}
```

Szenenbeleuchtung: Lichttypen

- Drei Lichttypen werden in VRML unterstützt (eigene Knotentypen)
- **Directional Light:**
 - parallel gerichtetes Licht einer unendlich weit entfernten Quelle
 - keine Abschwächung mit der Entfernung
- **PointLight:**
 - Licht breitet sich gleichmässig von punktförmiger Quelle aus (z.B. Glühlampe)
 - Abschwächung mit der Entfernung
- **SpotLight:**
 - Licht breitet sich kegelförmig von punktförmiger Quelle aus (z.B. Taschenlampe)
 - Abschwächung mit der Entfernung
- Wichtigste Feldtypen:
 - **direction**-Feld: Richtungsvektor
 - **ambientIntensity**-Feld: Stärke des Einflusses auf Objekte
 - **color**-Feld: Lichtfarbe
 - **location**-Feld: Position der Lichtquelle



Beispiel: Szene mit Beleuchtung

```
DirectionalLight {
  direction 0 -1.0 0
  ambientIntensity 0.7
  color 1.0 1.0 1.0
}

SpotLight {
  location -5.0 3.0 0
  direction 0.5 -0.5 -0.0
  ambientIntensity 0.4
  color 1.0 1.0 1.0
}

Group {
  ... wie letztes Beispiel
}

NavigationInfo {
  headlight FALSE
}
```

NavigationInfo

- Gibt globale Zusatzinformation für das Rendering an:
 - Z.B. Standardmodus
 - Z.B. Standardgeschwindigkeit
- Headlight:
 - Standardlichtquelle (directional) aus Betrachtersicht
 - kann in NavigationInfo ausgeschaltet werden

Strukturierung in VRML

- Wiederverwendung von Knoten:
 - `DEF Bezeichner Knoten`
 - `USE Bezeichner`
- Modularisierung:
 - `INLINE Datei`
 - » Bindet beliebige VRML-Datei an der gegebenen Stelle ein

Beispiel zu DEF/USE

```
Group {
  children [
    Transform {
      children [
        DEF RedBox Shape {
          appearance Appearance {
            material Material {
              diffuseColor 1.0 0 0
            }
          }
          geometry Box {
            size 2.0 2.0 2.0
          }
        }
      ]
      translation 2.0 0 0
    }
    ...
    Transform {
      children [
        USE RedBox
      ]
      translation -2.0 0 0
    }
  ]
}
```